1 Introduction

There are several cases in which we would like to compute the expectation of a function on a data set i.e. \( \mathbb{E}[f(X)] \). When \( X \) is a continuous random variable we cannot compute all the values of \( X \) so we use the proximity
\[
\mathbb{E}[f(X)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i).
\]
The problem we are facing now is how to sample \( x_1..x_n \) in an independent manner.

Apparently, this problem affects many different areas. Today we will see the Markov Chain Monte Carlo principle and two way of implementing it.

2 Probabilistic background

**Process:** A *Process* is a random variable or a set of random variables with different values in different times where time is a discrete variable. We’ll refer to these variables as \( X^{(1)}..X^{(T)} \). The probability of a process is the sum of the probabilities in all possible trajectories that could generate the process.

A *random process* is a process in which we cannot make any assumptions on how the process was generated. We will focus on Markov process.

**Markov process:** A *Markov process* is a process that follows the equation:
\[
\forall j < i \quad P(X^{(i)}|X^{(1)}..X^{(j)}) = P(X^{(i)}|X^{(j)})
\]

A Markov process hence is a process that has no memory, the value of \( X \) in a specific time depends only on the value it had in the time unit before with no relation to the value of \( X \) in the previous times. An example to a Markov process is a drunk man that walks in a room. Since he’s capable of walking in any direction with no relevance to the previous steps he took his next location depends solely on his position in the last time-unit.
Homogeneous process: A process will be called homogeneous if dynamics of moving from state $X^{(j)}$ to state $X^{(j+1)}$ is not time-dependent or if we’ll phrase it mathematically:

$$(\forall j < i, i' - j' = i - j) \quad P(X^{(i)} = s_2|X^{(j)} = s_1) = P(X'^{(i')} = s_2|X'^{(j')} = s_1)$$

The only thing that matters is the time difference between the two states. A process that is both Markov and homogeneous can be described by using a single probability $P(X^{(1)}|X^{(0)})$. We can write down these probability values as a matrix $p$ where $p_{st} = P(X^{(1)} = t|X^{(0)} = s)$, this matrix will be called transition matrix. In order to compute the probability of a state in a specific time unit we’ll have to know the starting probabilities as well, however, we’d like to manage without knowing these values.

A Markov chain can have an unreachable state. In order to avoid these situations we’ll demand that the Markov chains be irreducible. A state $j$ is considered to be accessible from state $i$ if $\exists n P(X^{(n)} = j|X^{(0)} = i) > 0$.

We’ll also discuss acyclic processes.

Acyclic process A state is called periodic if

$$\exists n \quad P(X^{(i+d)} = s|X^{(i)} = s) > 0 \iff d \mod n = 0$$

A process is called acyclic if for every state $s$, $s$ is not periodic.

Ergodic Markov chain A Markov chain will be called ergodic if it is homogeneous, acyclic and irreducible.

3 Stationary distribution

In an ergodic Markov chain it holds that $P(X^{(n)} = s|X^{(0)} = s_0)$ is not $s_0$ dependant for big enough $n$ or:

$$\forall s_0 \quad \lim_{n \to \infty} P(X^{(n)} = s|X^{(0)} = s_0) = \pi_s$$

(we will not prove this theorem in this lecture).

This distribution is called stationary distribution and the variable $\pi$ that we defined is a legal distribution and $\pi$ fulfills

$$\sum_s \pi_s = 1$$

Now it’s easy to sample the values of $\pi_s$ by starting in some value and sampling the next state according to the transition matrix over and over until we decide to stop. The big question now is when to stop? how many times do we need to sample from this transition matrix? There are chains that
are called mixing chains that promise that if we want an $\epsilon$ approximation to the real stationary distribution the number of steps we’ll need is polynomial of $\frac{1}{\epsilon}$ and $s$. Unfortunately, we will not be able to prove that the chains we will deal with are mixing chains.

A quite surprising example to ergodic Markov chain is the Monopoly game. The process is defined by being in a specific square in a specific turn of the game, the process is Markov since the only two parameters that control your square in time $n$ is your square in time $n-1$, the dice and specific some specific events, it’s acyclic and irreducible because you can reach every square and you can’t get into cycles. Surprisingly, the chance of stepping in Broadway is higher than the chance of stepping in Park Place.

### 3.1 $\pi_s$ features

Since $\pi_s$ is stationary we can conclude:

$$P(X^{(n+1)} = t) = \sum_s P(X^{(n)} = s) P(X^{(n+1)} = t|X^{(n)} = s) \tag{1}$$

and if we’ll take the limit $n \to \infty$ we’ll get

$$\pi_t = \sum_s \pi_s \text{p}_{st} \tag{2}$$

The stationary probability is the fixpoint of the system, if we look at the matrix $p$ and treat $\pi$ as a vector we can easily see that $\pi$ is an eigenvector with an eigenvalue of 1.

**Reversible Markov chains:** A reversible Markov chain is a process in which you can generate the same trajectory whether you walk forward or backward in the process. Another way of phrasing it is a process in which we can’t tell the order between states, if there are two given states we can’t tell which one came before the other. In reversible Markov chains:

$$P(X^{(i)} = s, X^{(i+1)} = t) = P(X^{(i)} = t, X^{(i+1)} = s) \tag{3}$$

and this is equal to:

$$\pi_s \text{p}_{st} = \pi_t \text{p}_{ts} \tag{4}$$

so the ratio between the stationary probability of two states is

$$\frac{\pi_s}{\pi_t} = \frac{\text{p}_{ts}}{\text{p}_{st}} \tag{5}$$

this property is called detailed balance.

The detailed balance feature is interesting in our problem since the posterior
is well defined but is hard to calculate numerically. For an example we’ll take the clustering problem. In this problem the posterior is defined as

$$\pi_\theta = P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

computing $P(D|\theta)$ is relatively easy and can be calculated as

$$P(D|\theta) = P(X_1[m]..X_n[m]|\theta) = \sum_c p(c|\theta) \prod_i P(X_i[m]|c, \theta)$$

and $P(\theta)$ is the prior probability. However, computing $P(D)$ is difficult since we need to integrate over all possible values of $\theta$

$$P(D) = \int_\theta P(D|\theta)P(\theta)$$

and this is difficult. Using the ratio from equation 5 we can compute the ratio between two posteriors:

$$\frac{\pi_{\theta_1}}{\pi_{\theta_2}} = \frac{P(D|\theta_1)}{P(D|\theta_2)} = \frac{p_{\theta_2,\theta_1}}{p_{\theta_1,\theta_2}} \tag{6}$$

if these states are reachable from each other ($p_{\theta,\theta_1} \neq 0$ and $p_{\theta,\theta_2} \neq 0$).

4 Building detailed balance systems

4.1 Metropolis-Hasting

Assuming we have a distribution $P(X) = f(X) \cdot \frac{1}{C}$ where $f(x)$ is known but the normalizing factor $\frac{1}{C}$ is unknown. We would like to build a detailed balance system and then we could sample from $P(X)$. The problem is that we need to compute the transition matrix $p$ and if the distribution is continuous this would be impossible. To overcome this obstacle we’ll use Metropolis-Hasting algorithm.

Metropolis first introduced the algorithm in 1949 and Hasting improved it in 1970. We’ll define a new distribution $r(s,t)$ that will be called the proposal distribution, the only restriction on this distribution is that it has to be symmetric i.e. $r(s,t) = r(t,s)$. Now that we have a proposal distribution we can compute $p_{st}$ as:

$$p_{st} = \begin{cases} 
    r(s,t) \frac{f(t)}{f(s)} & f(t) \leq f(s) \\
    r(s,t) & \text{else}
\end{cases} \tag{7}$$

this means that the transition probability from $s$ to $t$ will be the proposed probability if state $t$ is likelier than $s$ and if $s$ is likelier than $t$ we’ll multiply

4
the proposed probability in their probabilities ratio. Since we computed \( p_{st} \) like this we’ll get
\[
\frac{p_{st}}{p_{ts}} = \frac{r(s, t)}{r(t, s)} \frac{f(t)}{f(s)} = \frac{P(t)}{P(s)}
\] (8)
as in the detailed balance feature.
This will give us the Metropolis-Hasting algorithm:

**Sample-next-state(\( S \))**

\[
t = \text{sample from } r(s, t)
\]
\[
p = \text{sample uniformly from } [0, 1]
\]
\[
\text{if } p \leq \frac{f(t)}{f(s)} \text{ return } t
\]
\[
\text{else return } s
\]

This algorithm is similar to the transition probability \( p_{st} \) in equation 7. if \( f(t) \geq f(s) \) then \( \frac{f(t)}{f(s)} \geq 1 \) and \( p \) will surely be less than 1 so the new state will be \( t \) according to the proposal probability \( r(s, t) \). If \( f(t) < f(s) \) then the next state will be \( t \) according to the proposal probability \( r(s, t) \) multiplied by the probability of \( \frac{f(t)}{f(s)} \). We’ll notice that \( P(p < \frac{f(t)}{f(s)}) = \min(1, \frac{f(t)}{f(s)}) \).

The advantage of this method is that we need to compute the ratio \( \frac{f(t)}{f(s)} \) only for the states that we were advised to move to. The proposal probability can be built in different ways, it can give a high probability to move between close states. It can propose to move between distant states, a proposal which in most cases won’t be selected. We can try to find what proposal probabilities makes this chain mix rapidly.

### 4.2 Gibbs Sampler

Gibbs sampler was introduced by Geman & Geman in 1984. This algorithm constructs Markov chains in a manner that reminds the Metropolis-Hasting algorithm. In this algorithm \( p \) always = 1 and changing the state will be achieved by changing only one of the parameters of the state we’re at.

**Gibbs-sample(\( s = \{x_1..x_n\} \))**

\[
\text{set } i \leftarrow \text{sample index from some distribution } r_i
\]
\[
t = \{x'_1..x'_n\}
\]
\[ x'_j = x_j \text{ if } j \neq i \]

\[ x'_i \sim P(X_i|x_{1..i-1}, x_{i+1..n}) \]

return \( t \)

This algorithm produces detailed balanced Markov chains. Let’s assume that we changed the first index where \( s = \langle x_1..x_n \rangle \) and \( t = \langle x'_1, x_2..x_n \rangle \). The probability of getting from \( s \) to \( t \) i.e. \( p_{st} \) is

\[ p_{st} = r_1 p(x'_1|x_2..x_n) \]

Whereas getting from \( t \) to \( s \) will be

\[ p_{ts} = r_1 p(x_1|x_2..x_n) \]

Then their ratio will be

\[ \frac{p_{ts}}{p_{st}} = \frac{p(x_1|x_2..x_n) \cdot r_1}{p(x'_1|x_2..x_n) \cdot r_1} = \frac{p(x_1..x_n)}{p(x'_1, x_2..x_n)} = \frac{f(s)}{f(t)} \]

The Gibbs sampler is somewhat easier to implement than the Metropolis-Hasting algorithm since we don’t have to create the proposal distribution. Furthermore, computing \( p(x_1|x_2..x_n) \) can be done using the equation

\[ p(x_1|x_2..x_n) = \frac{p(x_1..x_n)}{\sum_{x'_1} p(x'_1, x_2..x_n)} \]

Most of the times the numerator and the denominator can be expressed as a product and then most of the elements will be reduced.

### 4.3 Back to the clustering problem

We will take the clustering problem as example to MCMC. In this problem the full distribution function will be

\[
P(\{C[m], X_1[m]..X_n[m]\}_{m=1}^M, \Theta_C, \Theta_{X_1|C}..\Theta_{X_n|C})
\]

\[
= P(\Theta_C) \prod_{i=1}^n P(\Theta_{X_i|C}) \prod_{m=1}^M \left[ P(C[m]|\Theta_C) \prod_{i=1}^n P(X_i[m]|C[m], \Theta_{X_i|C}) \right]
\]

We can split the data into three sets:

\[
\begin{align*}
D &= \{X_1[m]..X_n[m]\}_{m=1}^M \\
H_C &= \{C[m]\}_{m=1}^M \\
H_\Theta &= \Theta_C, \Theta_{X_1|C}..\Theta_{X_n|C}
\end{align*}
\]

Strategies we can take are:
1. Joint: compute $P(H_\Theta, H_C|D)$

2. Cluster: compute $P(H_C|D)$
   When the clusters are known we can compute $\Theta$ using expectation

3. Parameters: compute $P(H_\Theta|C)$
   Then we can compute the likelihood of every row being in a specific cluster.

In case we want to implement the first case we can use Gibbs sampler when we change the values of elements from $H_C$ and $H_\Theta$. In this case, sampling $P(C[1]|C[2]..C[m], H_\Theta, D)$ will be the probability of the first cluster given the observations of the first row and the parameters because the clusters are independent. Using this assumption we’ll get

$$= P(C[1]|X_1[1]..X_n[1], H_\Theta) \propto P(X_1[1]..X_n[1]|C[1], H_\Theta) P(C[1]|H_\Theta)$$

Changing $\Theta$ will be nastier, we need to compute

$$P(\Theta_C|\Theta_{X1[1]..X_m[C]} H_C, D) = P(\Theta_C|C[1]..C[m])$$

But we can make it easy by sampling for example from Dirichlet distribution. If we choose to implement case number 2 we’ll compute

$$P(H_C|D) = \int_{H_\Theta} P(H_C, H_\Theta|D)$$

$$= \frac{1}{P(D)} \int_{H_\Theta} P(H_C, H_\Theta, D)$$

$$= \frac{1}{P(D)} \int_{H_\Theta} P(D, H_C, H_\Theta) P(H_\Theta)$$

If we’ll change one of the clusters we’ll need to change the elements of $H_\Theta$ corresponding to the cluster we have changed therefor we’ll compute $\Theta_C$ for every possible change of the cluster then normalize it and that’s how we’ll compute that conditional distribution.

If we want to choose the third option we can use the ratio

$$\frac{P(\Theta|D)}{P(\Theta'|D)} = \frac{P(\Theta)P(D|\Theta)}{P(\Theta')P(D|\Theta')}$$

as an input for Metropolis-Hasting algorithm and run it on the data until we reach a steady state. However, running Gibbs sampler would be unpleasant because the conditional distribution will be hard to compute.
5 Running the simulation

Until now we calculated $p_{st}$ now we’ll discuss on how to run the simulation so it will give proper results.

The first and most simple method is called burn in. In this method we start from a random value and we run the simulation until we think we have a good mix (this is called the burn in period). After the burn in period we start sampling values every fixed number of steps.

The second strategy will be to run as many chains as the number of samples we want to sample, each chain will start from a random state and will be run $n$ iterations until a good mix will be achieved, when you get to that point you take the last state the chain was in.

The advantage of the “burn in” method is that you run a long run on the chain only once. The disadvantage is that the samples can be correlated. Another option is to run several number of chains and sample from them several times after the burn in period. Then we can compare the values that we got for the same states, if they converge to the same value then our sampling is good.

\[ 	ext{start} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
Figure 2: Using M different chains